\*\*\*Topic B - Simplifying Code Using XAML\*\*\*

\\ C# version

var toolbar = new StackPanel();

toolbar.Orientation = Orientation.Horizontal;

var newButton = new Button();

newButton.Content = "New";

newButton.Background = new SolidColorBrush(Colors.Pink);

toolbar.Children.Add(newButton);

var openButton = new Button();

openButton.Content = "Open";

openButton.Background = new SolidColorBrush(Colors.Pink);

toolbar.Children.Add(openButton);

\\ XAML version

<StackPanel Name="toolbar" Orientation="Horizontal">

<Button Name="newButton" Background="Pink">New</Button>

<Button Name="OpenButton" Background="Pink">Open</Button>

</StackPanel>

\*\*\*Topic C - Creating an App for Universal Windows Platform\*\*\*

\\ Markup for the Grid element

<Button Margin="6" Padding="6" Name="clickMeButton">

Click Me

</Button>

\\ Modify the XAML to wrap the Button element inside a horizontally orientated StackPanel that is inside a vertically orientated StackPanel

<StackPanel>

<StackPanel Orientation="Horizontal">

<Button Margin="6" Padding="6" Name="clickMeButton">

Click Me

</Button>

</StackPanel>

</StackPanel>

\\ Modify the Button element to give it a new event handler for its Click event

<Button Margin="6" Padding="6" Name="clickMeButton"

Click="clickMeButton\_Click">

Click Me

</Button>

\*\*\*Topic D - Sharing Resources\*\*\*

\\ Add the following markup inside the existing <Application> element in the App.xaml file

<Application.Resources>

<LinearGradientBrush x:Key="rainbow">

<GradientStop Color="Red" Offset="0" />

<GradientStop Color="Orange" Offset="0.1" />

<GradientStop Color="Yellow" Offset="0.3" />

<GradientStop Color="Green" Offset="0.5" />

<GradientStop Color="Blue" Offset="0.7" />

<GradientStop Color="Indigo" Offset="0.9" />

<GradientStop Color="Violet" Offset="1" />

</LinearGradientBrush>

</Application.Resources>

\*\*\*Topic D - Replacing a Control Template\*\*\*

\\ In the App.xaml file, add the following markup inside the <Application.Resources> element

<ControlTemplate x:Key="DarkGlassButton" TargetType="Button">

<Border BorderBrush="#FFFFFFFF"

BorderThickness="1,1,1,1" CornerRadius="4,4,4,4">

<Border x:Name="border" Background="#7F000000"

BorderBrush="#FF000000" BorderThickness="1,1,1,1"

CornerRadius="4,4,4,4">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Border Opacity="0" HorizontalAlignment="Stretch"

x:Name="glow" Width="Auto" Grid.RowSpan="2"

CornerRadius="4,4,4,4">

</Border>

<ContentPresenter HorizontalAlignment="Center"

VerticalAlignment="Center" Width="Auto"

Grid.RowSpan="2" Padding="4"/>

<Border HorizontalAlignment="Stretch" Margin="0,0,0,0"

x:Name="shine" Width="Auto"

CornerRadius="4,4,0,0">

<Border.Background>

<LinearGradientBrush EndPoint="0.5,0.9"

StartPoint="0.5,0.03">

<GradientStop Color="#99FFFFFF" Offset="0"/>

<GradientStop Color="#33FFFFFF" Offset="1"/>

</LinearGradientBrush>

</Border.Background>

</Border>

</Grid>

</Border>

</Border>

</ControlTemplate>

<Style TargetType="Button">

<Setter Property="Template"

Value="{StaticResource DarkGlassButton}" />

<Setter Property="Foreground" Value="White" />

</Style>

\*\*\*Topic E - Binding to Elements\*\*\*

\\ In the MainWindow.xaml file, add the following markup after the Button element inside the horizontally orientated StackPanel

<Slider Value="50" Maximum="100" Minimum="0"

Width="200" Name="slider"/>

<TextBlock Text="{Binding ElementName=slider, Path=Value}"

VerticalAlignment="Center" Margin="10"/>

\\ Under the horizontally orientated StackPanel, but inside the outer StackPanel, add the following markup to define some instructions to the user, a slider for values between 0 and 360 degrees, and a red square with a rotation transformation

<TextBlock>Use the slider below to rotate the square:</TextBlock>

<Slider Value="0" Minimum="0" Maximum="360"

Name="sliderRotation"/>

<Rectangle Height="100" Width="100" Fill="Red">

<Rectangle.RenderTransform>

<RotateTransform

Angle="{Binding ElementName=sliderRotation, Path=Value}" />

</Rectangle.RenderTransform>

</Rectangle>

\*\*\*Topic E - Binding to Data\*\*\*

\\ Right-click on the Models folder and add a new class named Employee and add the following statements to it

public class Employee

{

public int EmployeeID { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public DateTime DOB { get; set; }

public decimal Salary { get; set; }

}

\\ Add another class named EmployeesViewModel

public class EmployeesViewModel

{

public HashSet<Employee> Employees { get; set; }

public EmployeesViewModel()

{

Employees = new HashSet<Employee>();

Employees.Add(new Employee

{ EmployeeID = 1, FirstName = "Alice", LastName = "Smith",

DOB = new DateTime(1972, 1, 27), Salary = 34000M });

Employees.Add(new Employee

{ EmployeeID = 2, FirstName = "Bob", LastName = "Jones",

DOB = new DateTime(1965, 4, 13), Salary = 64000M });

}

}

\\ Add the following ListBox element whose items are bound to each employee instance in the Employees hash set of the view model in the MainPage.xaml file

<ListBox ItemsSource="{Binding Employees}">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal">

<TextBlock Text="{Binding EmployeeID}" FontSize="30" />

<TextBox Text="{Binding FirstName}"

Header="First Name" Margin="10" />

<TextBox Text="{Binding LastName}"

Header="Last Name" Margin="10" />

<DatePicker Date="{Binding DOB}"

Header="DOB" Margin="10" />

<TextBox Text="{Binding Salary}"

Header="Salary" Margin="10" />

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

\*\*\*Topic F - Animating with Storyboards\*\*\*

\\ Open the MainPage.xaml file, change Grid into Canvas, and add an ellipse to make a red ball

<Canvas Background=

"{ThemeResource ApplicationPageBackgroundThemeBrush}">

<Ellipse Fill="Red" Height="100" Width="100"/>

</Canvas>

\\ Modify the squashing effect by changing the value of ScaleY to be 0.666 (that is, 66.6%) of its normal height at time position 1s, and return to exactly 1 at time position 1.2s:

<DoubleAnimationUsingKeyFrames Storyboard.TargetProperty=

"(UIElement.RenderTransform).(CompositeTransform.ScaleY)"

Storyboard.TargetName="ellipse">

<EasingDoubleKeyFrame KeyTime="0:0:0.8" Value="1"/>

<EasingDoubleKeyFrame KeyTime="0:0:1" Value="0.666"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.2" Value="1"/>

</DoubleAnimationUsingKeyFrames>

\\ In the toolbox, choose Button and draw one on the canvas named BounceBallButton, change its contents to Bounce Ball, and give it a Click event handler

<Button Name="BounceBallButton" Content="Bounce Ball"

Canvas.Left="154" Canvas.Top="45"

Click="BounceBallButton\_Click"/>